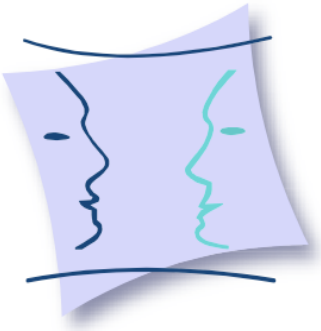
Technische Universität Darmstadt



Telecooperation

Prof. Dr. Max Mühlhäuser

Distributed Execution of S-BPM Business Processes

October 2010

Dr. Erwin Aitenbichler, **Stephan Borgert**

Telecooperation Group

Department of Computer Science, Technische Universität Darmstadt, Germany.

Outline

1) Introduction
2) Related Work
3) Mapping of PASS Processes to CCS
4) Process Execution
5) Implementation
6) Conclusion

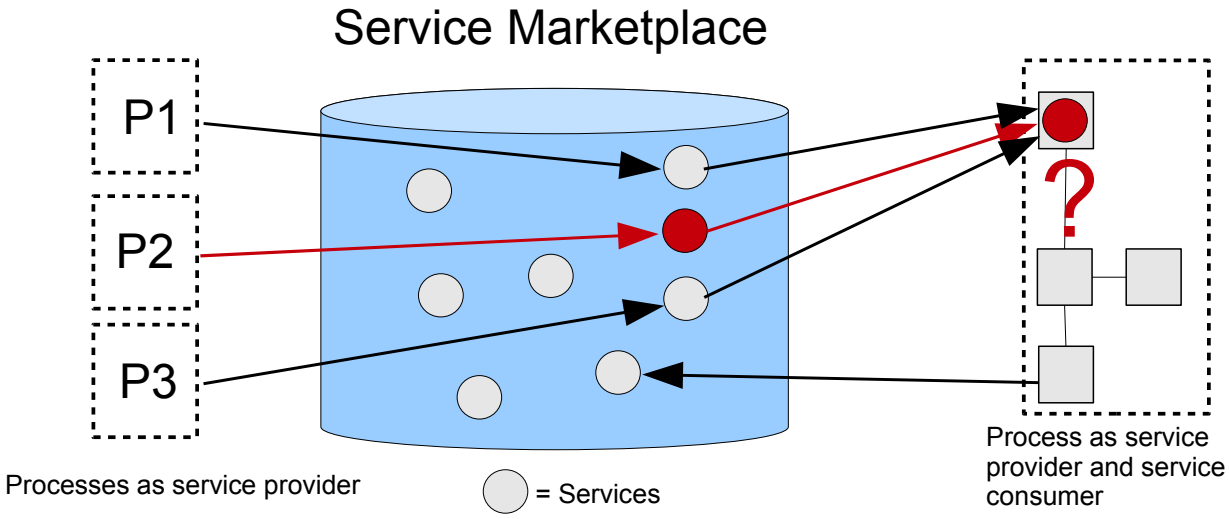
1)Introduction
Motivation

- Internet-based marketplaces for mobile phone end-user applications (“apps”) have been very successful recently.
- Increased adoption of service-oriented architectures in enterprises.
  - make systems modular
  - components become interchangeable

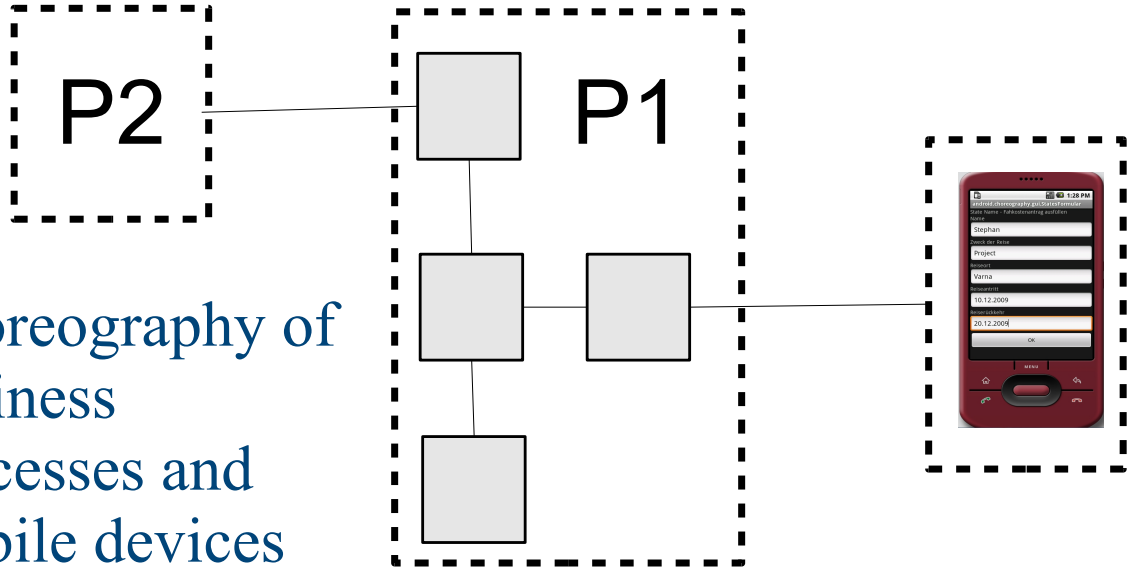
Combination leads to IoS (Internet of Services)

Open service marketplaces will lead to multiple services with same functionality but differences in many details.

==> Compatibility check is necessary



1) Introduction



Choreography of business processes and mobile devices

- Every process participant may use their own process execution engine.
- These engines may be used by different enterprises, but there may also be multiple engines within the same enterprise.
- The engine may also run on the mobile device of an employee.

==> Distributed execution of processes is necessary

1) Introduction

This work addresses the following two aspects of distributed process execution:

- the composition-time compatibility check of services and
- the execution-time message routing between process engines

Needed technologies

1. suitable process modeling techniques. It should be possible to
   - clearly split up a process into subprocesses describing the work of each process participant separately.
   - perform the same verification methods for processes that are modeled in a whole and processes that are modeled in a distributed manner.
2. flexible communication platform

1) Introduction

In this work, we make the following contributions:

- Show that PASS (Parallel Activities Specification Scheme) fulfills the requirements for distributed modeling and execution of business processes.
- Describe a process embedding concept and its implementation
- Show that pub / sub abstraction is a good basis for implementation

2) Related Work

Process description languages

Mainstream process description languages like EPC, BPMN lack a

- formal foundation
- well defined semantics

- ===> formal verification of the models or direct execution of the models is impossible

Execution oriented languages like BPEL, XPDL

- hardly allow the description of choreographies
- lack formal verification techniques
- are often hard to use for businessmen

2) Related Work

Process description languages

Transformations from the process layer to the application layer

- are often hard to realize (e.g. BPMN -> BPEL)
- often lead to deviations of the process model from the executed process

BPEL extensions

- mostly do not support the description of choreographies.

\+ and / or services provided by humans are not considered

- lack of formal verification techniques

2) Related Work

Formal Process Description Methods

Petri nets

- widely accepted as formal foundation for business process modeling
- providing a graphical and easily understandable notation

+ but one main drawback is that the entire process has to be modeled in a single net and parallelism is not directly supported

Process Algebras

- inherently support choreography modeling, refining and clustering
- provide a rich theory for formal verification

+ but one main drawback is the textual representation which is harder to understand than graphical representations

2) Related Work

Formal Process Description Methods

Parallel Activities Specification Scheme (PASS)

- is used in the Metasonic BPM Suite
- provides a graphical and easily understandable notation
- supports choreography modeling
- supports services that are performed by humans
- ==> it fulfills our requirements

Stephan Borgert
Prof. Dr. M. Mühlhäuser
Telekooperation
©

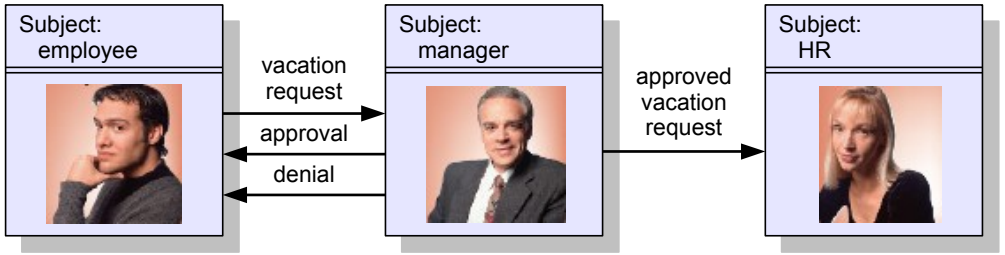
3) Mapping of PASS Processes to CCS

PASS as implementation of CCS

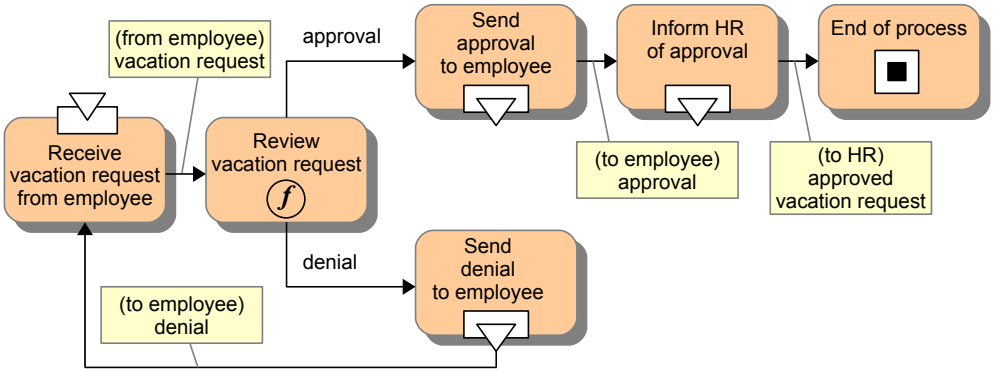
Processes are modeled on two different levels

- The subject interaction level where the message exchange is modeled
- Level of internal behavior of the subjects

subject interaction level



internal behavior of the subject manager



3) Mapping of PASS Processes to CCS

How to verify the process models ?

- PASS is not formal founded yet
  - no formal verification methods are implemented yet

  + therefore we mapped PASS to CCS (Calculus of Communicating Systems)

  + and used an available third party tool(CWB-NC) for verifications

Why did we choose CCS and no other formalism ?

- PASS was inspired by CCS
- ==> mapping is simple

+ and tools for CCS are already available

3) Mapping of PASS Processes to CCS

Brief introduction into CCS

Syntax of CCS: Let $\mathcal{L}$ be a set of labels and let $Act = \{\tau\} \cup \mathcal{L} \cup \{\bar{a} | a \in \mathcal{L}\}$ be the set of all actions. Then the syntax of process P is given by:

$$P ::= 0 \;\Big|\; A \;\Big|\; a.P \;\Big|\; \sum_{i \in I} P_i \;\Big|\; P_1 | P_2 \;\Big|\; P \backslash L \;\Big|\; P[f]$$

with $a \in Act$, $L \subseteq \mathcal{L}$, $f : \mathcal{L} \to \mathcal{L}$, and $P, P_i \in \mathcal{P}$ where $\mathcal{P}$ is the set of processes. 0 is the inactive process that does nothing.

- CCS knows three different types of actions
  - Send
  - Receive
  - tau. tau is defined as an internal action which is invisible from the outside.

3) Mapping of PASS Processes to CCS

Semantic of CCS: The semantic of the CCS operators is given by rules of the type:

$$\frac{Pre_i, \ldots, Pre_n}{Imp}$$

- The Prei are the premises that are met if the implication Imp is satisfied.
- If n = 0 then there are no premises and Imp always holds.
- This kind of rules are called Structural Operational Semantic rules and were introduced by Plotkin.

3) Mapping of PASS Processes to CCS

Name	Sym.	Structural Operational Semantic
action	.	$\frac{}{a.P \xrightarrow{a} P}$
process identifier	:=	$\frac{P \xrightarrow{a} P'}{A \xrightarrow{a} P'} \; if \; A := P$
choice	$\sum$	$\frac{P_j \xrightarrow{a} P'_j}{\sum_{i \in I} P_i \xrightarrow{a} P'_j} \; j \in I$
parallel composition	|	$\frac{P \xrightarrow{a} P'}{P \mid Q \xrightarrow{a} P' \mid Q} \quad \frac{Q \xrightarrow{a} Q'}{P \mid Q \xrightarrow{a} P \mid Q'} \quad \frac{P \xrightarrow{a} P', \; Q \xrightarrow{\bar{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$
restriction	$\backslash$	$\frac{P \xrightarrow{a} P'}{P \backslash L \xrightarrow{a} P' \backslash L} \; a, \bar{a} \notin \mathcal{L}$
renaming	f	$\frac{P \xrightarrow{a} P'}{P[f] \xrightarrow{f(a)} P'[f]} \; where f(\tau) = \tau, f(\bar{a}) = \overline{f(a)}$

Formal semantic of CCS is defined by SOS rules

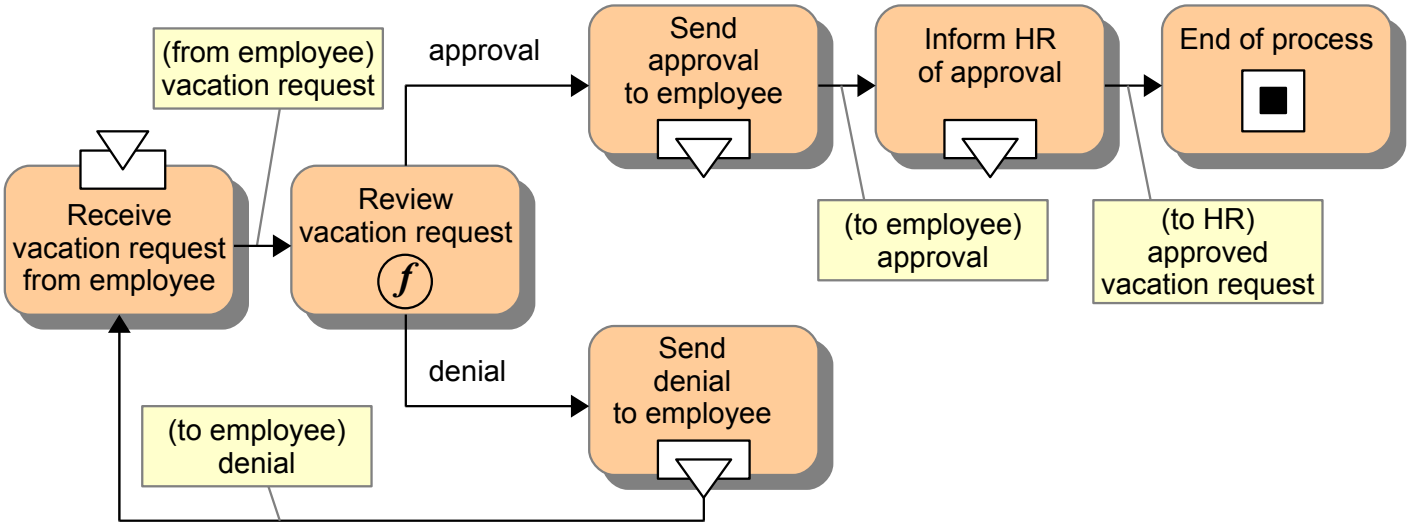
Stephan Borgert
Prof. Dr. M. Mühlhäuser
Telekooperation
©

3) Mapping of PASS Processes to CCS

Different types of activities

Activity Types in PASS:

- Send
- Receive
- Function



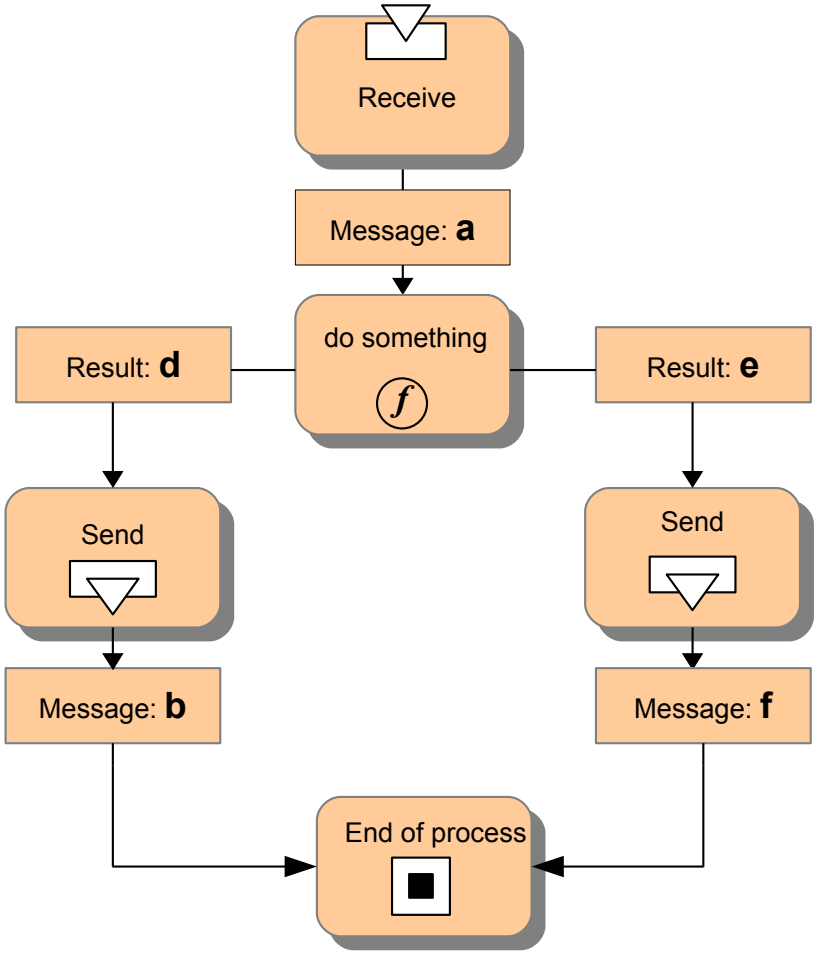
Activity Types in CCS:

Let $\mathcal{L}$ be a set of labels then
$Act = \mathcal{L} \cup \{\bar{a} | a \in \mathcal{L}\} \cup \{\tau\}$
is the set of all CCS actions.

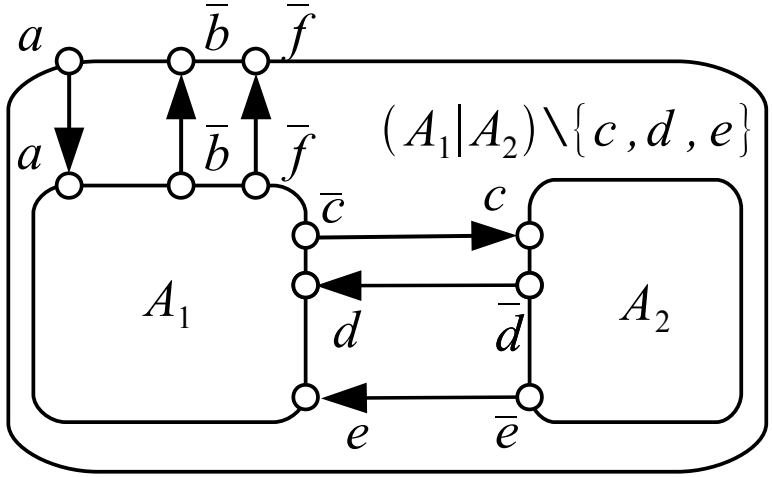
Stephan Borgert
Prof. Dr. M. Mühlhäuser
Telekooperation
©

3) Mapping of PASS Processes to CCS

Modeling internal behavior with CCS



subject A modeled with PASS

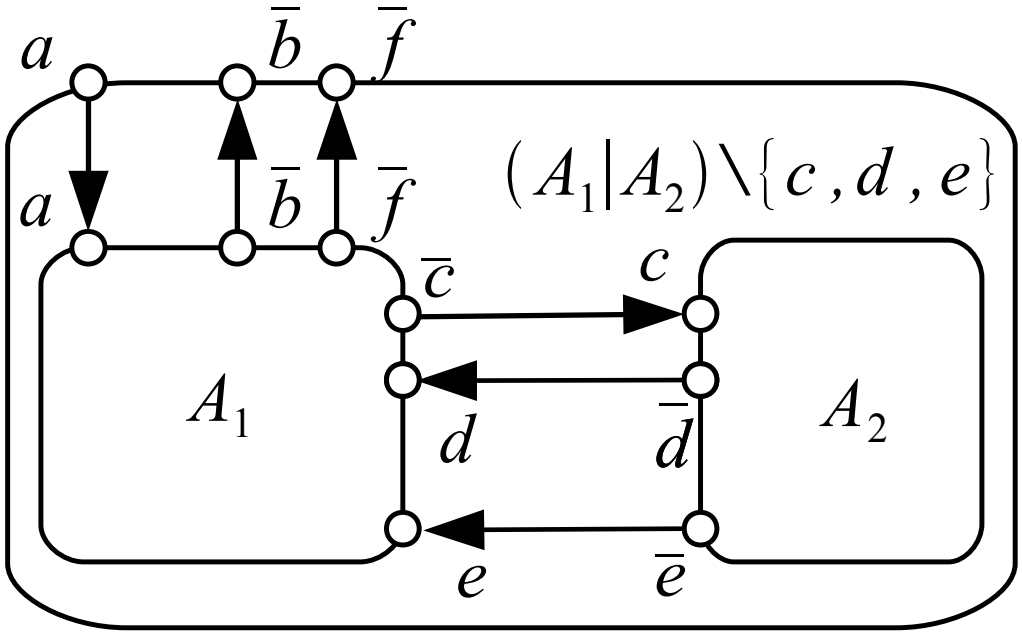


$$A_1 := a.\bar{c}.(d.\bar{b}.0 + e.\bar{f}.0)$$
$$A_2 := c.(\bar{d}.A_2 + \bar{e}.A_2)$$
$$A := (A_1|A_2)\backslash\{c,d,e\}$$

subject A modeled with CCS

3) Mapping of PASS Processes to CCS

Modeling internal behavior with CCS



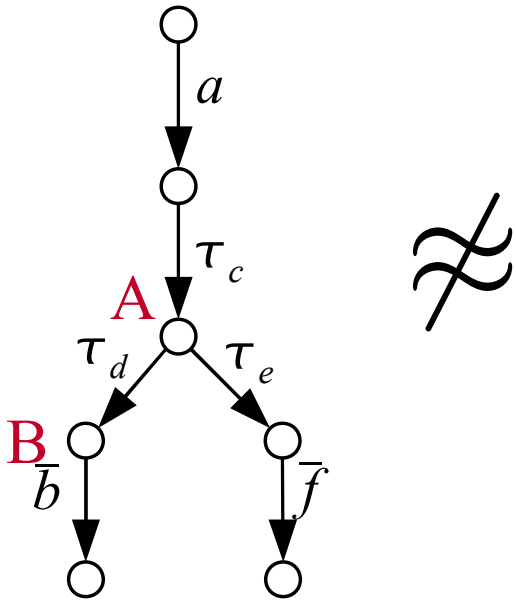
$$A_1 := a.\bar{c}.(d.\bar{b}.0 + e.\bar{f}.0)$$
$$A_2 := c.(\bar{d}.A_2 + \bar{e}.A_2)$$
$$\begin{aligned} A &:= (A_1 \mid A_2)\backslash\{c,d,e\} \\ &= (a.\bar{c}.(d.\bar{b}.0 + e.\bar{f}.0) \mid c.(\bar{d}.A_2 + \bar{e}.A_2))\backslash\{c,d,e\} \\ &= a.(\bar{c}.(d.\bar{b}.0 + e.\bar{f}.0) \mid c.(\bar{d}.A_2 + \bar{e}.A_2))\backslash\{c,d,e\} \\ &= a.\tau_c.((d.\bar{b}.0 + e.\bar{f}.0) \mid (\bar{d}.A_2 + \bar{e}.A_2))\backslash\{c,d,e\} \\ &= a.\tau_c.(\tau_d.(\bar{b}.0 \mid A_2) + \tau_e.(\bar{f}.0 \mid A_2))\backslash\{c,d,e\} \end{aligned}$$
$$A = a.\tau_c.(\tau_d.\bar{b}.(0 \mid A_2) + \tau_e.\bar{f}.(0 \mid A_2))\backslash\{c,d,e\}$$

$$\Longrightarrow A \approx a.\tau_c.(\tau_d.\bar{b}.0 + \tau_e.\bar{f}.0)$$

3) Mapping of PASS Processes to CCS

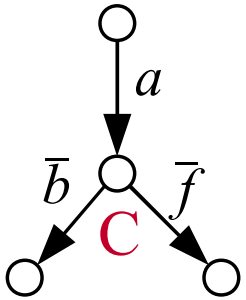
Why not remove all τ ?

$a.\tau_c.(\tau_d.\bar{b}.0+\tau_e.\bar{f}.0)$



$\not\approx$

$a.(\bar{b}.0+\bar{f}.0)$



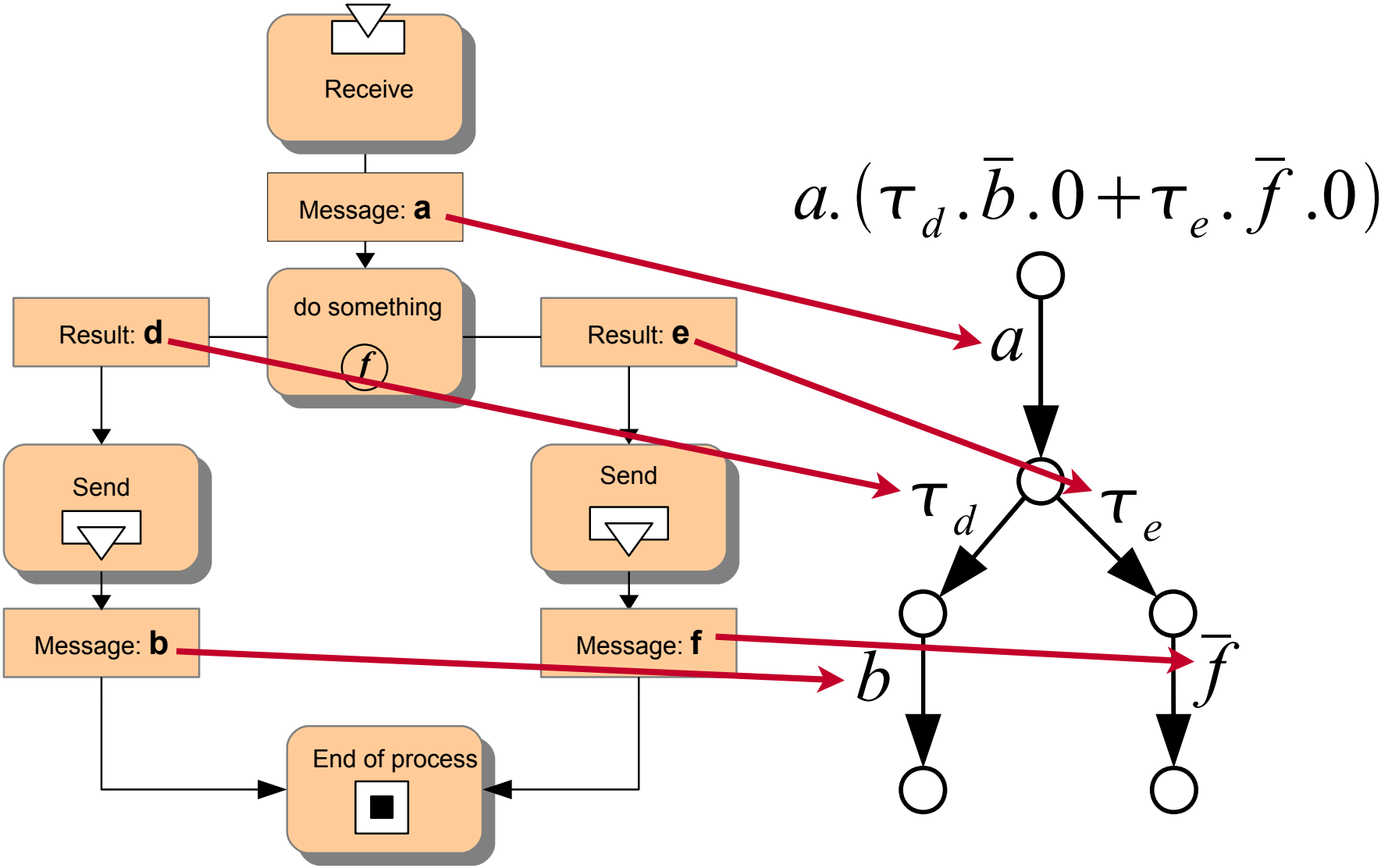
but for practical purposes:

$A = a.\tau_c.(\tau_d.\bar{b}.0 + \tau_e.\bar{f}.0) \longrightarrow a.(\tau_d.\bar{b}.0 + \tau_e.\bar{f}.0)$

Stephan Borgert
Prof. Dr. M. Mühlhäuser
Telekooperation
©

3) Mapping of PASS Processes to CCS

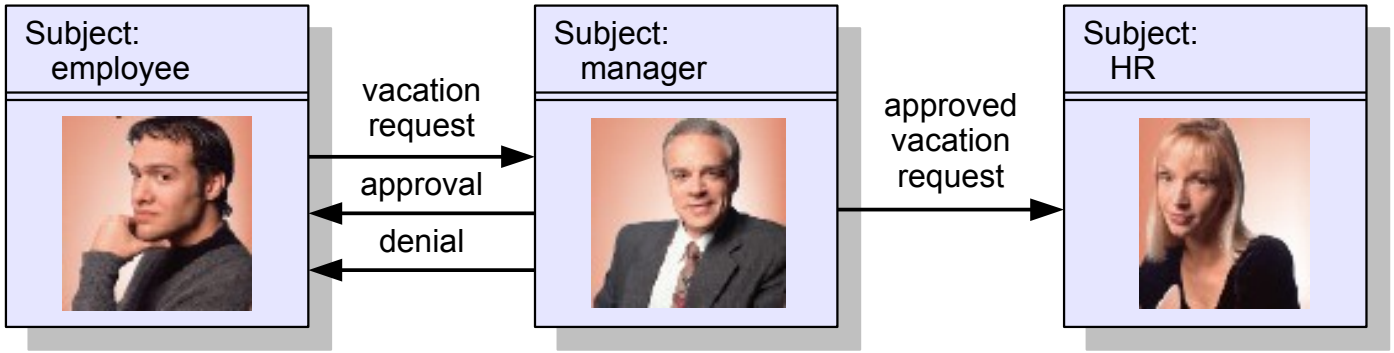
Final mapping of the internal behavior to CCS



3) Mapping of PASS Processes to CCS

Mapping of whole processes to CCS

PASS model



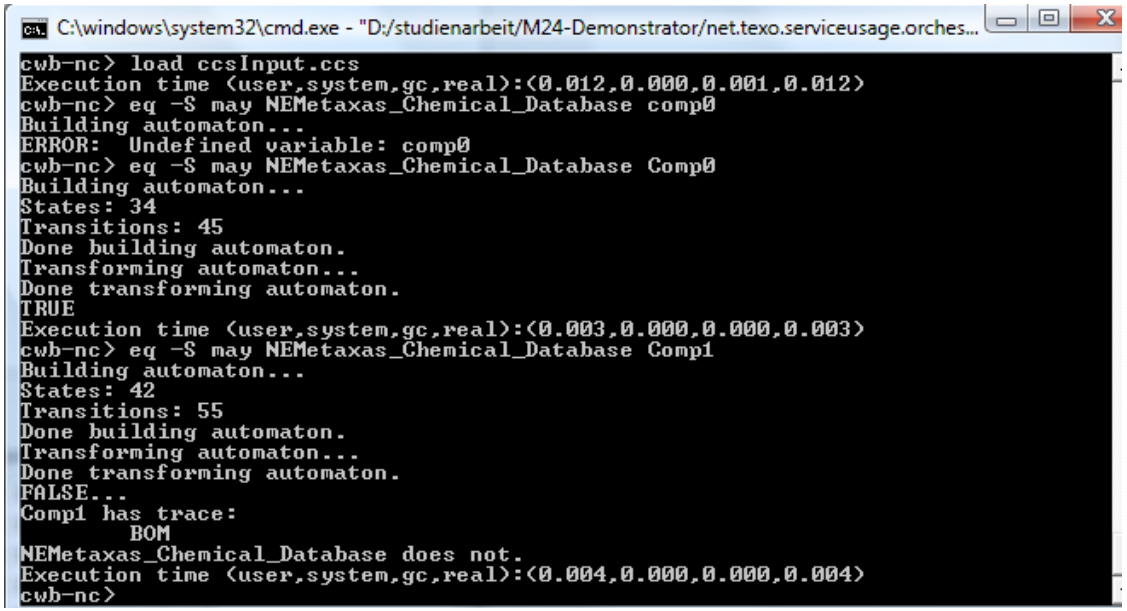
CCS model

$$P_1 := (employee|manager|HR)$$

$$P := P_1 \backslash \{vacaction\_request, approval, denial, approved\_vacation\_request\}$$

3) Mapping of PASS Processes to CCS

Verifying Process Compatibility



CWB-NC (Concurrency Workbench of the New Century)

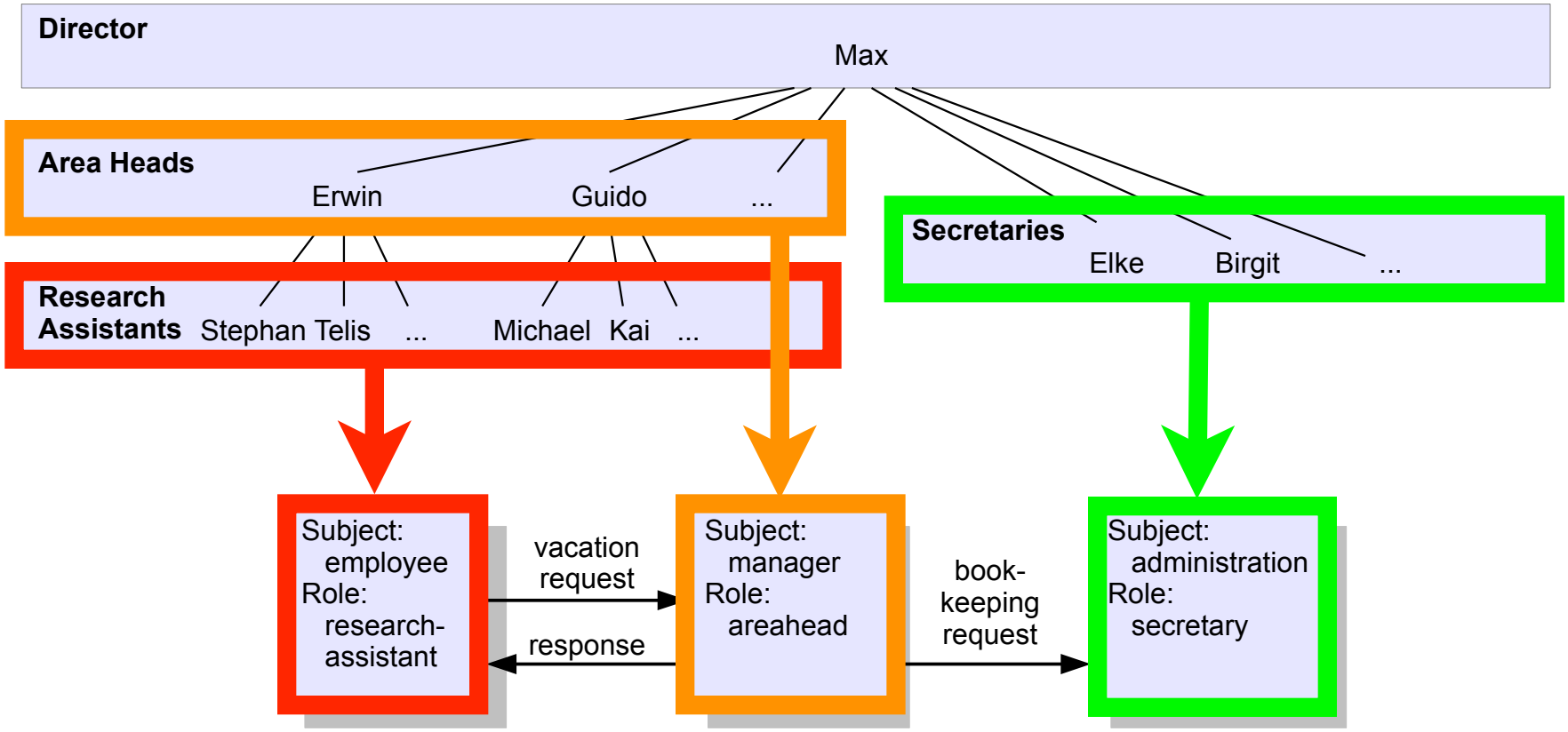
- supports various behavioral equivalences
- model checks
  - properties are written in mu-calculus

Allows to perform

- choreography conformance check
- reachability analysis to ensure that all end states can be reached

4) Process Execution

Process embedding

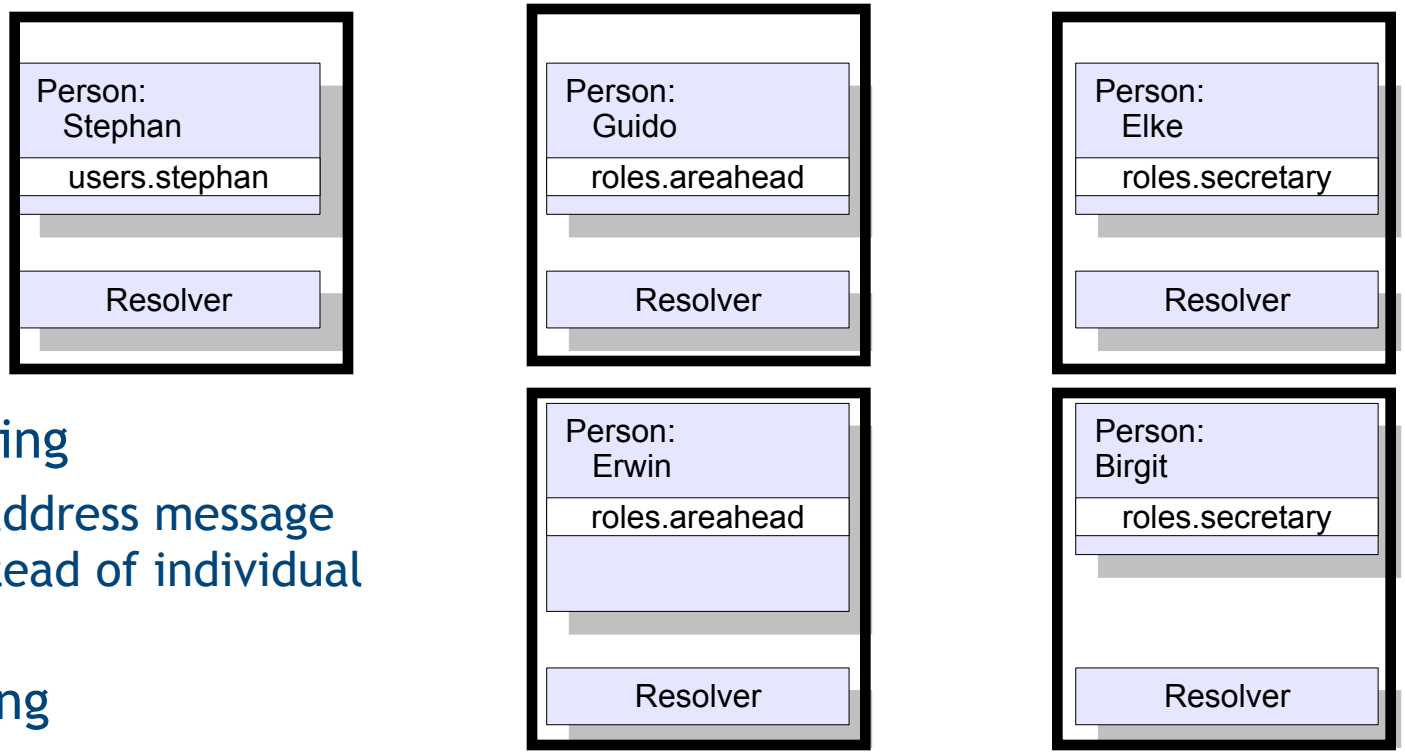


assigning of roles to subjects
e.g. subject manager is handled by area head

embedding information is modeled separately

4) Process Execution

Publish / Subscribe: System



Space decoupling

- producers address message brokers instead of individual consumers

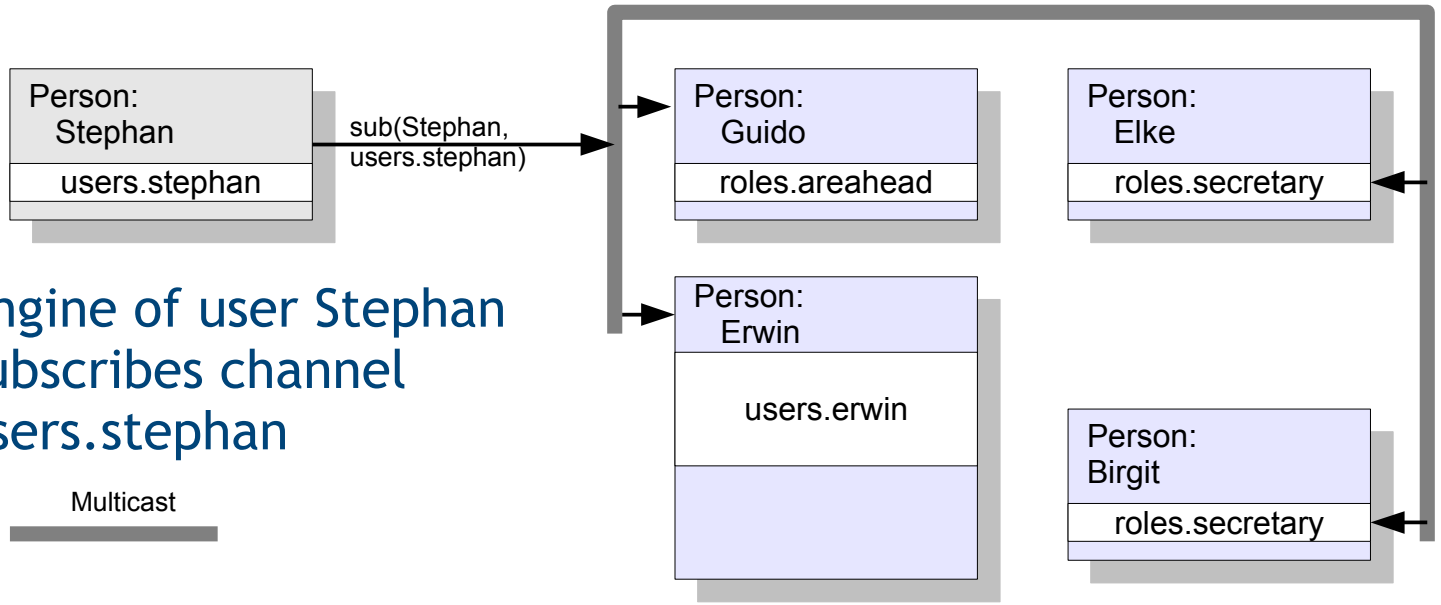
Time decoupling

- producers and consumers do not need to actively participate at the same time

- Each subject provider uses its own process engine
- Each process engine is equipped with a context resolver service

4) Process Execution

Publish / Subscribe: subscription



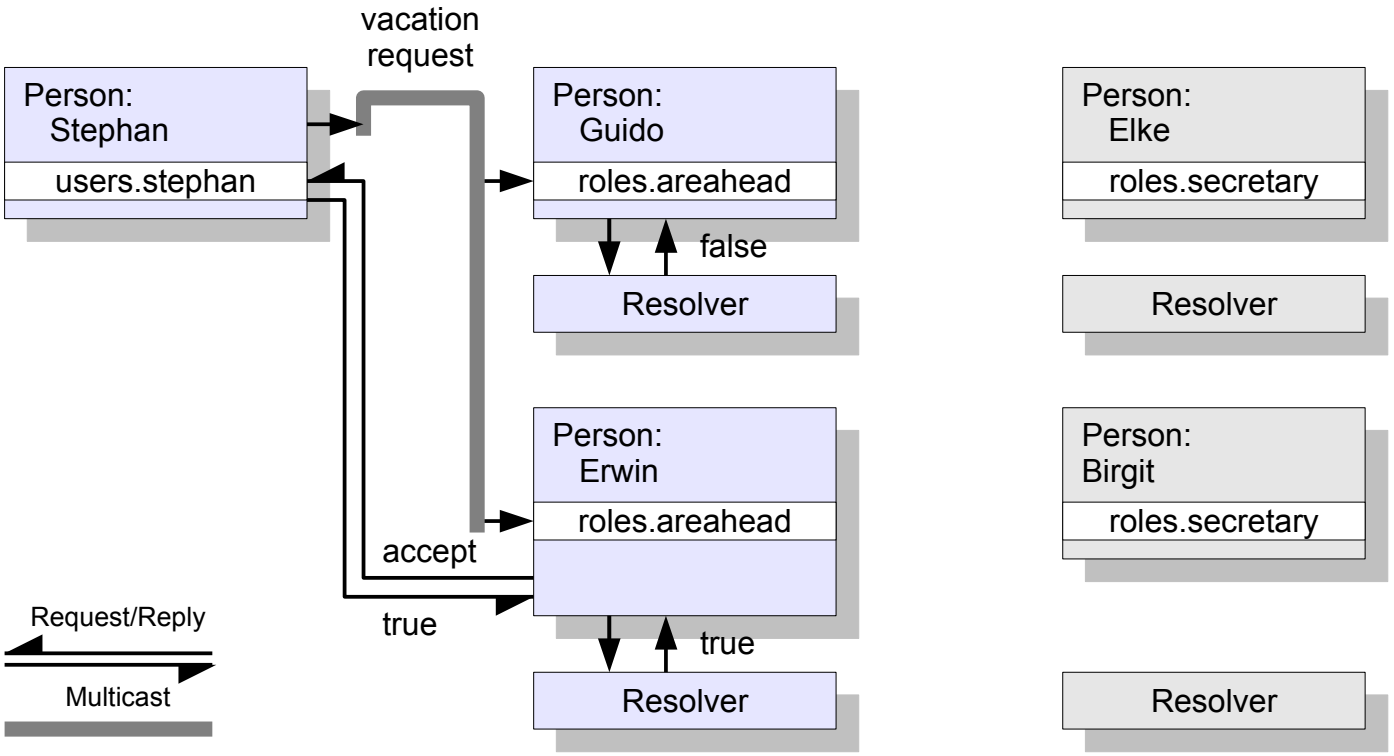
engine of user Stephan subscribes channel users.stephan

sub(Stephan, users.stephan)
sub(Erwin, users.erwin)
sub(Elke, users.elke)

sub(Stephan, roles.employee)
sub(Erwin, roles.areahead)
sub(Elke, roles.secretary)

4) Process Execution

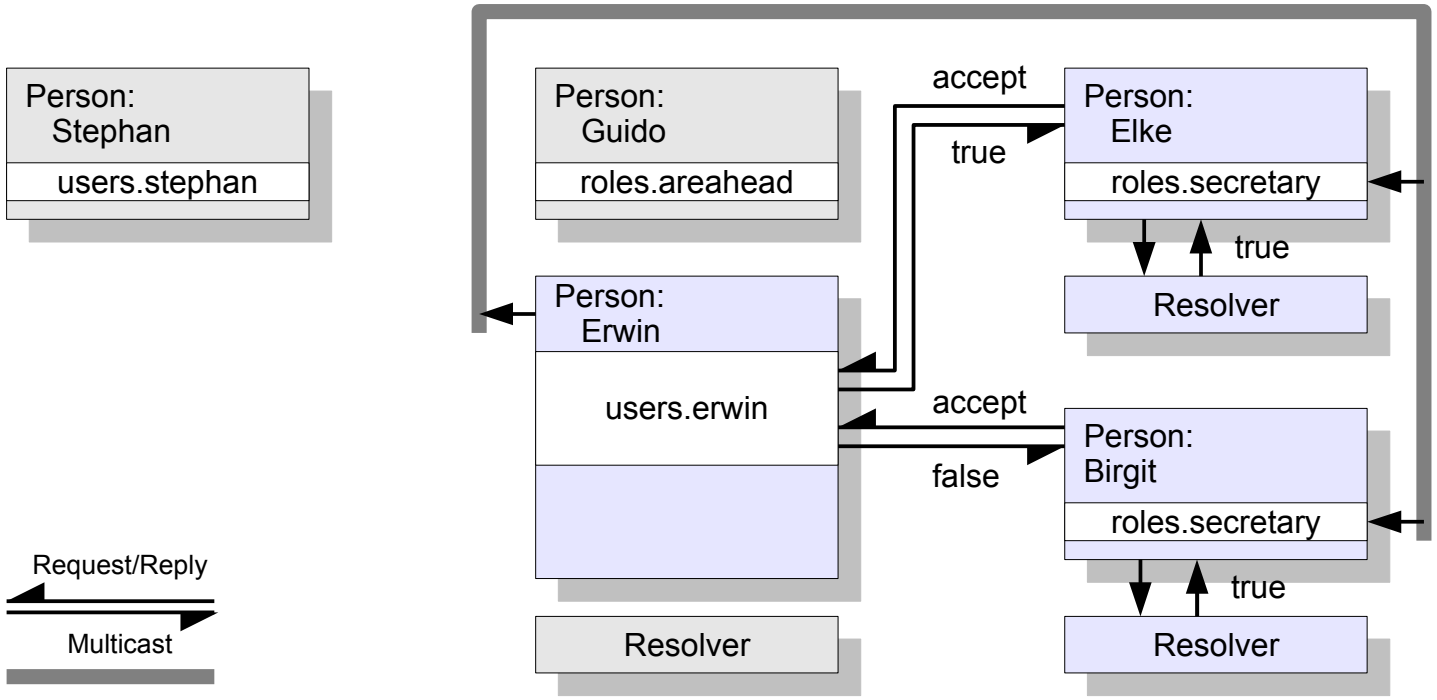
Publish / Subscribe: Instantiation



Selection of a subject provider is based on context

4) Process Execution

Publish / Subscribe:



Selection among subject providers with equal status (anycast)

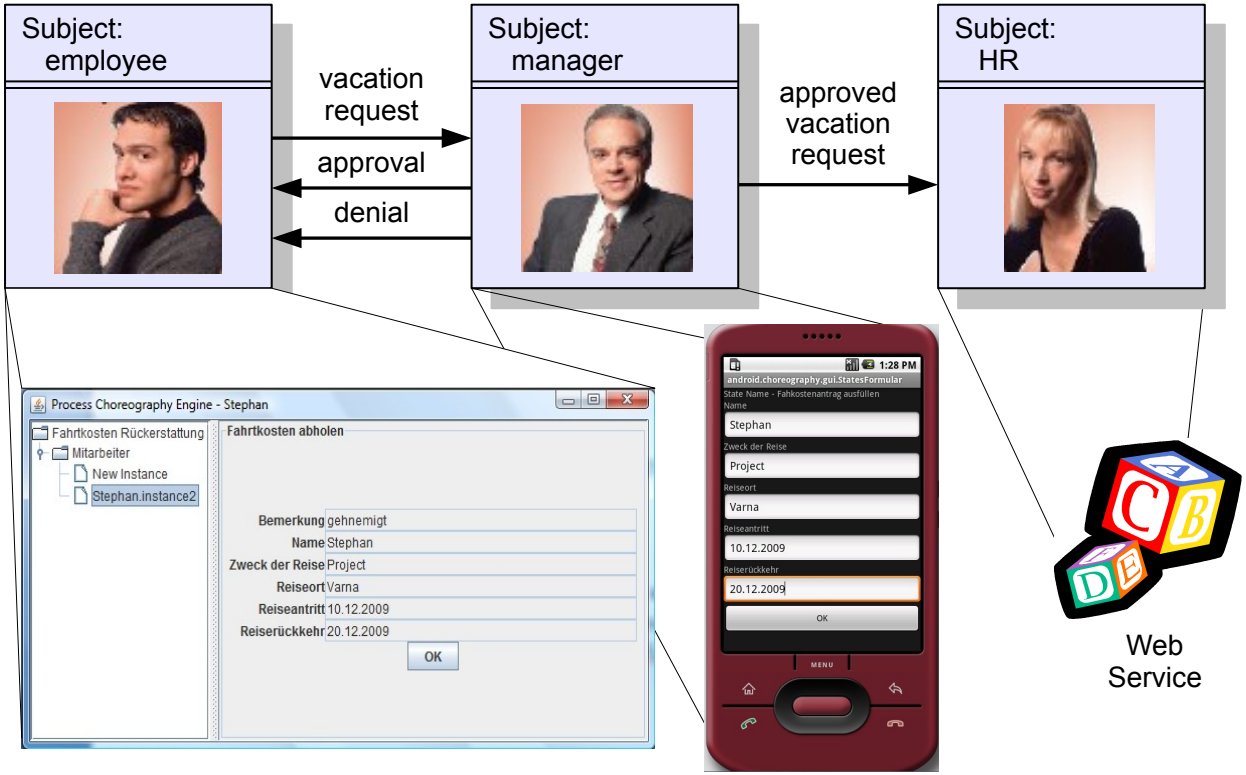
5) Implementation

MundoCore

- To put it in a nutshell...
  - MundoCore is an integration framework
  - It supports ad-hoc peer discovery and
  - builds a peer-to-peer publish/subscribe system on top
- MundoCore itself is service-oriented
  - The service model is similar to „fast web services“
  - Bridges exist to XML/SOAP, RMI, and CORBA
- MundoCore was designed for ubiquitous computing
  - supports very „small systems“
  - minimum footprint of middleware about 42 KB
  - supported programming languages: Java, C++, .net (, Python, JS)
  - supported platforms: Java (SE, J2ME/CDC, J2ME/CLDC), Win32, WinCE, Mac OS/X (Std, iPhone/iPod), Linux, ucLinux, SunOS
- Available as open-source from Telecooperation Group, Technische Universität Darmstadt

5) Implementation

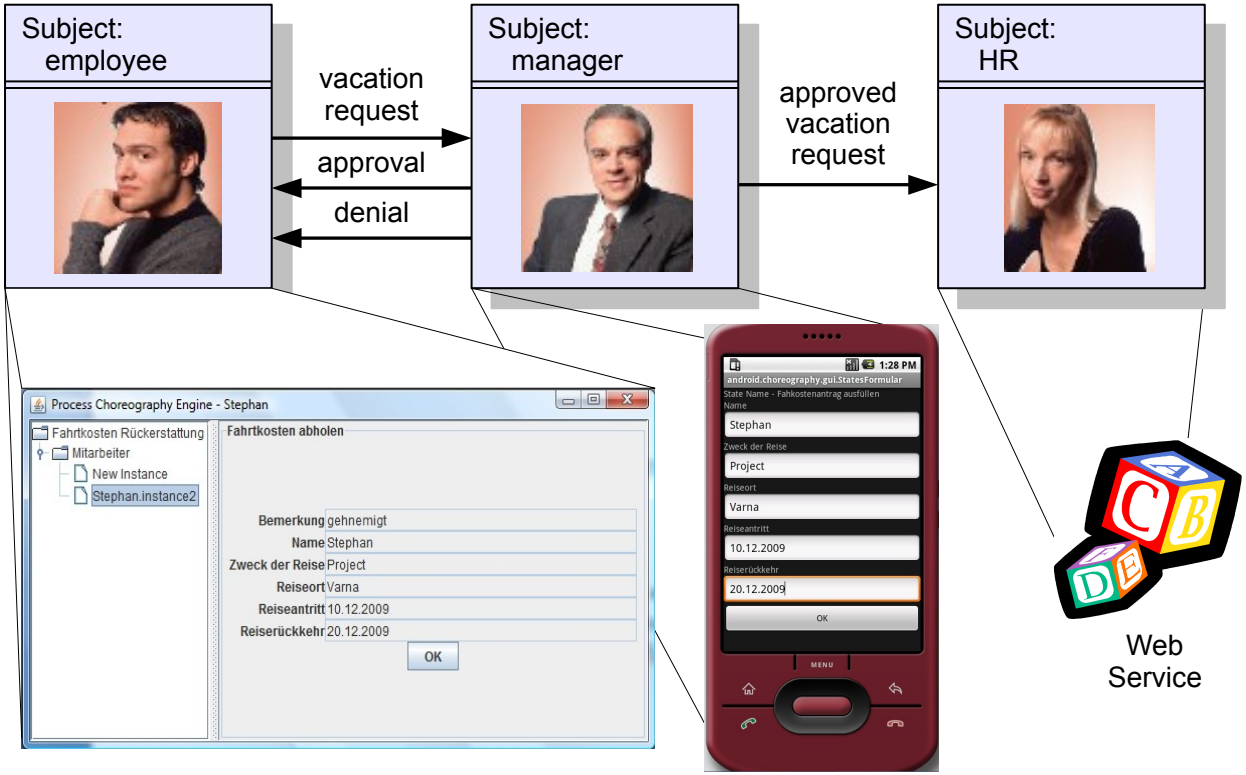
Process execution engine 1/ 2



- engine executes models created with the jPASS modeling tool
- each started engine executes a subset of the subjects defined in the process

5) Implementation

Process execution engine 2/ 2



- through interconnection of several engines, the whole process can be executed
- the engine supports PCs and mobile devices, i.e., Android based phones

6) Conclusion

- We have described an approach for the distributed execution of business processes, based on
  - subject-oriented process modeling
  - a Publish/Subscribe communication middleware
- the system provides several benefits in two important use cases:

6) Conclusion

Use Case 1) Splitting up process models along its subjects and executing the parts in a distributed manner.

- this allows to execute processes on servers of different departments
- employees can execute processes on their mobile devices and work on processes even when they do not have Internet connectivity

Use Case 2) If cooperating processes are modeled in a distributed manner, which is often the case when different businesses cooperate,

- then we can still perform the same model checks as if the overall choreography was modeled as a single, centralized process.
- In addition, the businesses do not have to disclose their detailed internal processes.

Thank you for your attention!

Stephan Borgert - stephan@tk.informatik.tu-darmstadt.de
Phd Student at Telecooperation Group
Department of Computer Science, Technische Universität Darmstadt, Germany.

For further questions about MundoCore do not hesitate to contact:

Dr. Erwin Aitenbichler - erwin@tk.informatik.tu-darmstadt.de